(12) UK Patent Application (19) GB (11) 2 357 600 (13) A

(43) Date of A Publication 27.06.2001

(21) Application No 9930283.8

(22) Date of Filing 23.12.1999

(71) Applicant(s)
International Business Machines Corporation
(Incorporated in USA - New York)
Armonk, New York 10504, United States of America

(72) Inventor(s)
Ray Cowey
Sharon McLoone
Jim O'Shea
Brian Walsh

(74) Agent and/or Address for Service
C J Ling
IBM United Kingdom Limited, Hursley Park,
WINCHESTER, Hampshire, SO21 2JN,
United Kingdom

(51) INT CL[7]
G06F 9/445

(52) UK CL (Edition S)
G4A AFL

(56) Documents Cited
GB 2329052 A US 5894571 A

(58) Field of Search
UK CL (Edition R) G4A AFL
INT CL[7] G06F 9/445
ONLINE: EPODOC, WPI, JAPIO, ELSEVIER, INSPEC, TDB

(54) Abstract Title
Hardware dependent software installation

(57) A method of installing software associated with hardware installed in a personal computer comprises the steps of providing one or more unique identifiers in the hardware, interrogating the unique identifiers, responsive to the interrogation, identifying the software associated with the installed hardware and installing said associated software. A computer software installation system is claimed with means for interrogating the unique identifier, means for identifying associated software and means for installing said software. A computer program product is also claimed for installing hardware-associated software by the above method. The unique identifiers may be configuration registers and may comprise information relating to the vendor, device type and revision level.

GB 2 357 600 A

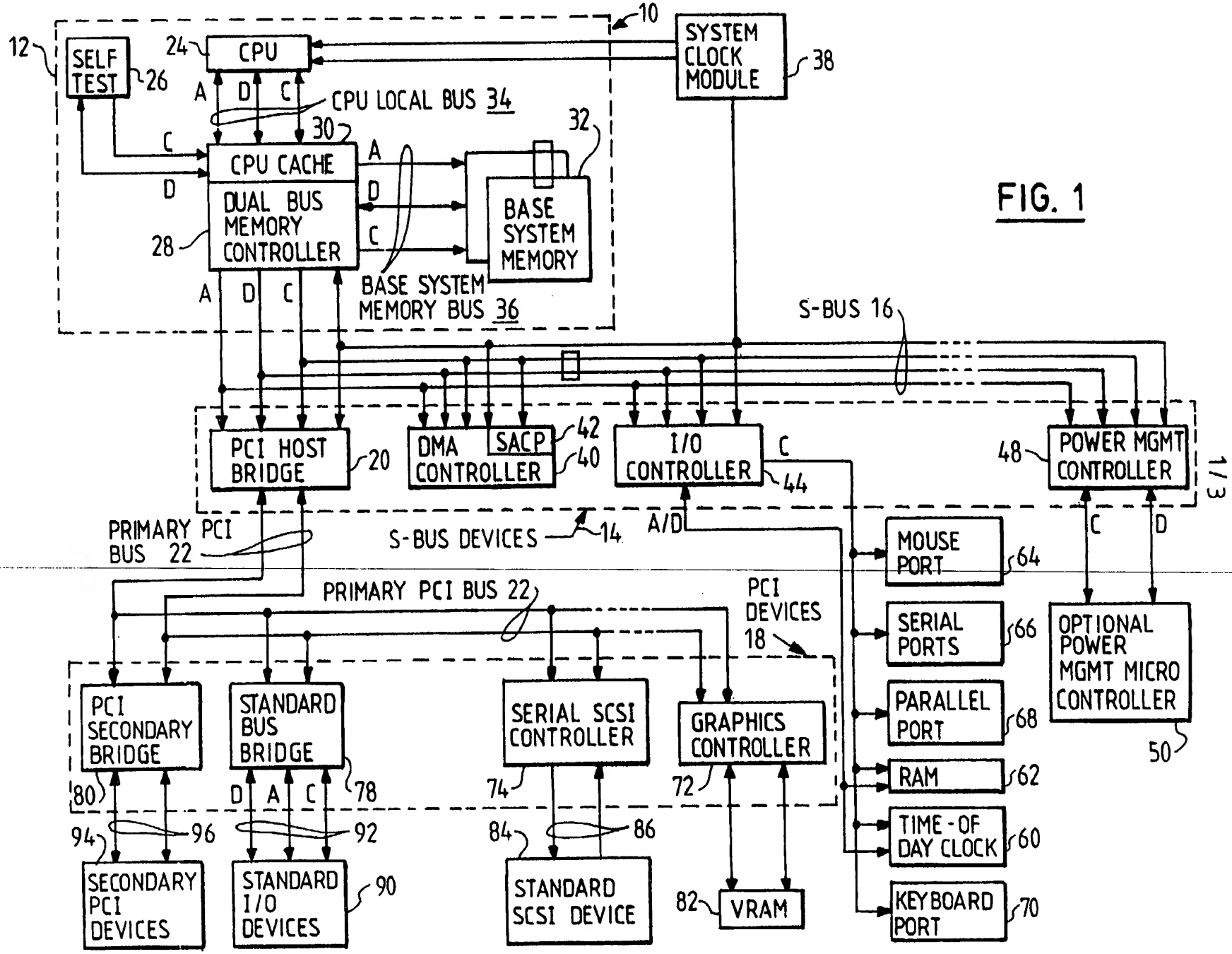
10
12
SELF TEST
26
24
CPU
A
D
C
CPU LOCAL BUS 34
30
C
D
CPU CACHE
DUAL BUS MEMORY CONTROLLER
28
A
D
C
BASE SYSTEM MEMORY BUS 36
32
BASE SYSTEM MEMORY
SYSTEM CLOCK MODULE
38
S-BUS 16
PCI HOST BRIDGE
20
DMA CONTROLLER
SACP
42
40
I/O CONTROLLER
C
44
48
POWER MGMT CONTROLLER
C
D
A/D
S-BUS DEVICES
14
PRIMARY PCI BUS 22
PRIMARY PCI BUS 22
PCI DEVICES 18
MOUSE PORT
64
SERIAL PORTS
66
PARALLEL PORT
68
RAM
62
TIME-OF DAY CLOCK
60
KEYBOARD PORT
70
OPTIONAL POWER MGMT MICRO CONTROLLER
50
PCI SECONDARY BRIDGE
80
STANDARD BUS BRIDGE
D
A
C
78
SERIAL SCSI CONTROLLER
74
GRAPHICS CONTROLLER
72
94
96
92
84
86
SECONDARY PCI DEVICES
STANDARD I/O DEVICES
90
STANDARD SCSI DEVICE
82
VRAM

FIG. 1

1/3

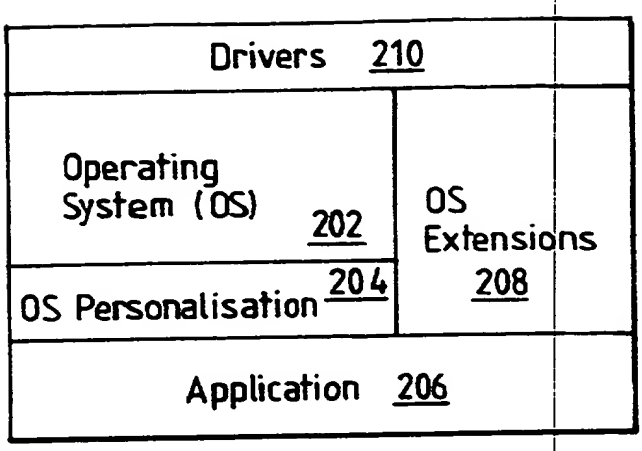
Drivers 210
Operating System (OS) 202
OS Extensions 208
OS Personalisation 204
Application 206

FIG. 2

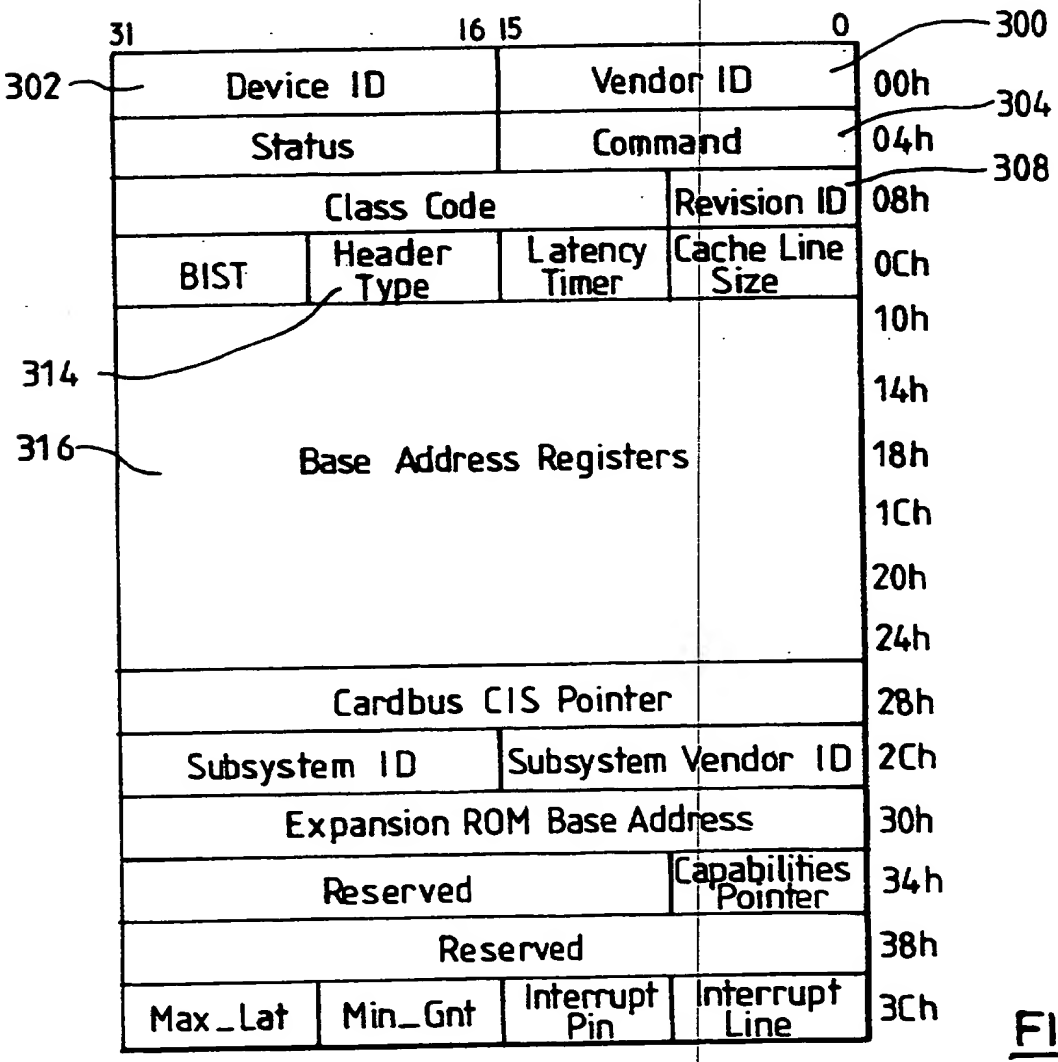
31
16 15
0
300
302
Device ID
Vendor ID
00h
304
Status
Command
04h
308
Class Code
Revision ID
08h
BIST
Header Type
Latency Timer
Cache Line Size
0Ch
10h
314
14h
316
Base Address Registers
18h
1Ch
20h
24h
Cardbus CIS Pointer
28h
Subsystem ID
Subsystem Vendor ID
2Ch
Expansion ROM Base Address
30h
Reserved
Capabilities Pointer
34h
Reserved
38h
Max_Lat
Min_Gnt
Interrupt Pin
Interrupt Line
3Ch

FIG. 3

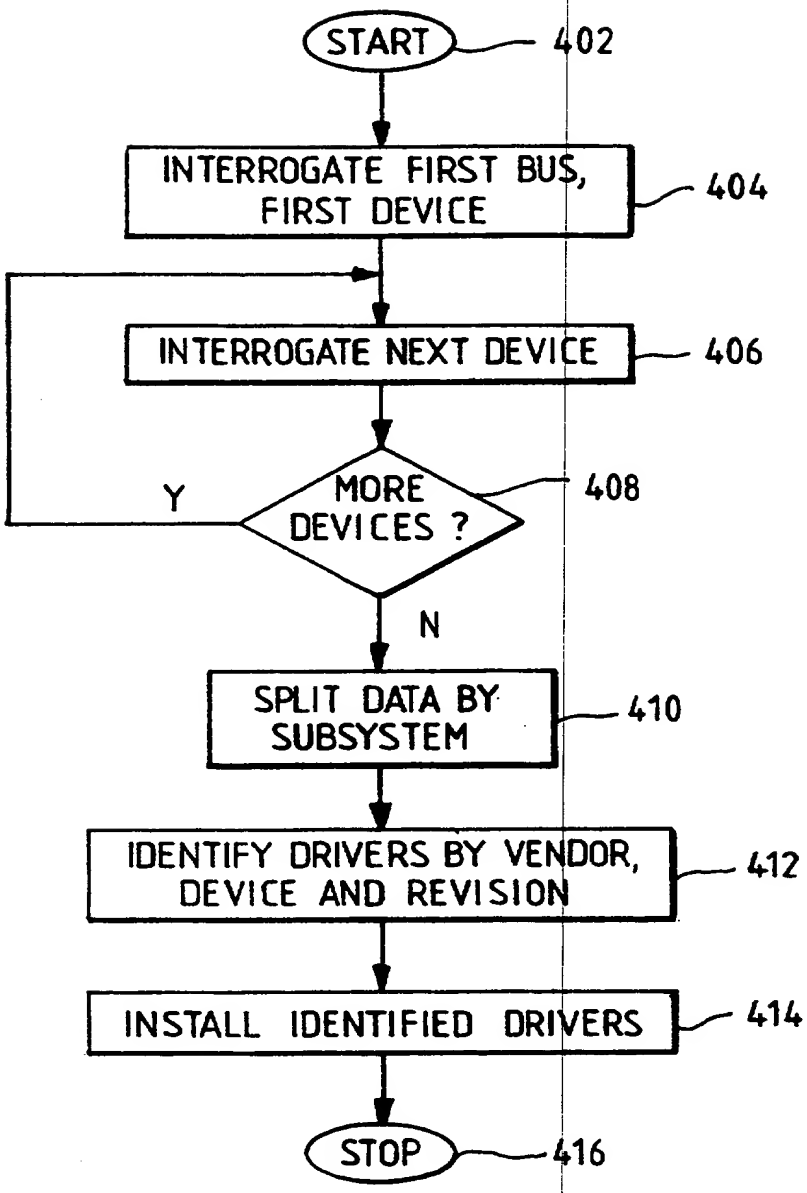
START
402
INTERROGATE FIRST BUS, FIRST DEVICE
404
INTERROGATE NEXT DEVICE
406
MORE DEVICES ?
408
Y
N
SPLIT DATA BY SUBSYSTEM
410
IDENTIFY DRIVERS BY VENDOR, DEVICE AND REVISION
412
INSTALL IDENTIFIED DRIVERS
414
STOP
416

FIG. 4

HARDWARE DEPENDENT SOFTWARE INSTALLATION

Field of the Invention

The present invention relates to installation of software on personal computers and more particularly to the installation of software customised to the hardware features of a particular personal computer.

Background of the Invention

When a personal computer is ordered, it is necessary that an operating system is installed on the personal computer in order to enable the computer to function and provide the user with a user friendly environment. The operating system is specific to each individual personal computer as it must be personalised to suit the hardware configuration by the installation of driver software. When many identical machines having the same operating system are being built, the installation of the operating system and personalised driver software may be implemented more quickly using a "pre-load". A pre-load comprises an operating system which itself consists of kernel code and a graphical user interface plus additional drivers for the hardware that is installed in that machine. The pre-load is generated by installation of the operating system on a machine of the type to be built. An image copy of the installed operating system is then taken from that machine and is used to "clone" copies of the installed operating system. The pre-load may optionally include other software such as application software which is to be included on the personal computer. A pre-load can only be used to cover a small number of pre-defined hardware configurations.

Another known solution is to generate an image to be installed on a particular system based on the customer's order form or on a manufacturing bill of material. This has the disadvantage that the drivers installed may not match the hardware installed in the personal computer. If the customer order has been updated since the order form was completed or if there is an updated version of the personalised hardware or the software driver, then this will not be reflected in the pre-loaded image.

A current trend in the manufacture of personal computers is for more specific hardware configurations such as modem cards, network cards, video and audio cards and the like to be available in personal computers, these hardware configurations being installed by the manufacturer. This proliferation of potentially unique hardware configurations forces a similar proliferation of potentially unique

operating systems with the associated drivers to enable any hardware additions to be operational. This type of customer driven environment also requires a rapid manufacturing turn around time from receipt of the customer's order until shipment of the personal computer from the manufacturing plant. The conventional method of building customer's images (manual installation of the operating system, adding drivers for any associated hardware additions) is not able to support a rapid turnaround time. The customer image may have an order quantity of one personal computer type with a potential selection of numerous different hardware additions.

PCT Patent Application WO94/08288 discloses a system in which a boot image for a particular personal computer is generated at an initial boot time by that computer based on the hardware present. That boot image is stored on the personal computer. When the computer is subsequently booted, the hardware present is checked to see if it has changed, and if it has not, then the previously stored boot image is loaded. This system does not address the generation of a boot image with hardware-dependent software, such as drivers, selected from a range of different software, such as drivers, for that particular personal computer.

PCT Patent Application WO95/17714 discloses a system in which a specific software object is used based on the identification number of the computer.

US Patent 5,794,032 discloses a system for automatic identification and configuration of computer peripherals for use at initialisation time in a personal computer. The system uses a program which is run every time the personal computer is turned on. The program uses a query instruction to obtain information as to the type of device attached to a particular interface. The query instruction is part of the "Attention Application Programming Interface" (ATAPI) and is only applicable to devices such as mass storage devices which support such an interface and not to other computer devices in general.

US Patent 5,717,930 discloses a method of installing an operating system program on a computer in a network. A bar code is read by a bar code reader. The bar code contains installation information relevant to the particular personal computer. Based on the content of the bar code, software information stored in an installation server on the network is read and installed in the personal computer. This method requires the use of a bar code reader attached to the personal computer. Additionally the bar code that is generated may not conform to the particular hardware configuration of

the computer on which the software is to be installed. If the customer order has been updated since the bar code was generated or if there is an updated version of the personalised hardware or the software driver, then this will not be reflected in the bar code.

Standard pre-loads can only cover expected configurations. So it would be desirable to provide a method of installing operating system and driver software that always reflects the actual configuration of the personal computer and always reflects the correct level of the driver software.

Disclosure of the Invention

Accordingly, the present invention provides a method of installing software associated with hardware installed in a computer, the method comprising the steps of: providing one or more unique identifiers in the hardware; interrogating the unique identifiers; responsive to the interrogation, identifying the software associated with the installed hardware; and installing the associated software.

This invention allows customisation to cover unexpected hardware configurations that may be requested by customers. Methods of software installation from the customer order are difficult and expensive to maintain in an environment where many new options are becoming available and software driver levels are frequently changing. This invention allows the software configuration to be computed at the latest possible point in the system manufacturing process and so automatically takes into account any late changes in configuration or late driver updates.

Preferably, the unique identifiers are configuration registers.

Further preferably the configuration registers comprise information relating to the vendor, device type and revision level.

Yet further preferably, the associated software is identified by means of vendor, device type and revision level information.

The invention also provides a computer software installation system for the installation of software associated with one or more pieces of hardware installed in a computer, Each piece of hardware having a unique identifier, the system comprising: means for interrogating the unique identifier; means for identifying which software is associated with the particular piece of hardware; and means for installing the software corresponding to the particular piece of hardware onto the computer.

The invention further provides a computer program product for use in a data processing system having a non-volatile storage medium, the computer program product comprising: a computer usable medium having computer readable program code means embodied in said medium for installing software associated with hardware installed in a computer, each piece of hardware having one or more unique identifiers in the hardware, said computer program product having: computer readable program code means for interrogating the unique identifiers; computer readable program code means, responsive to the computer readable program code means for interrogation, for identifying the software associated with the installed hardware; and computer readable program code means for installing the associated software.

Brief Description of the Drawings

Embodiments of the invention will now be described, by way of example, with reference to the accompanying drawings, in which:

Figure 1 is a block diagram of a prior art personal computer;

Figure 2 is a block diagram of the software portions to be installed onto the prior art personal computer of figure 1 using the installation method of the present invention;

Figure 3 is a map of the configuration space provided by configuration registers in each of PCI devices of Figure 2; and

Figure 4 is a flowchart of the installation method of the present invention.

Detailed Description of the Invention

Referring now to Figure 1, a multi-bus information handling system is shown generally at 10, comprising, (i) a processor, cache and memory complex 12 connected to S-bus (system bus) devices 14 via an S-bus 16 and (ii) primary PCI devices 18 attached to one of the S-bus devices, a primary PCI host bridge 20, via a primary PCI bus 22. More detailed descriptions of the processor, cache and memory complex 12, the S-bus devices 14, the primary PCI devices 18, and the other elements shown in Figure 1 will be provided hereinafter.

The processor, cache and memory complex 12 comprises a central processing unit (CPU) 24, a self-test circuit 26, a memory controller 28, a CPU cache 30, and base system memory 32. The CPU 24 in the preferred embodiment is a Pentium III microprocessor available from Intel Corp., although it is contemplated that the system 10 may be

implemented using other types of CPUs, especially other x86-type microprocessors (Pentium and Intel are trademarks on Intel Corp.). The self-test circuit 26 provides a built-in-self-test (BIST) feature for the CPU 24 upon power-up. The self-test circuit also controls any self-test features which may be incorporated within each of the S-bus devices 14.

The CPU 24 is connected to the self-test circuit 26 and the memory controller 28 by a CPU local bus 34. The memory controller 28 is connected to the base system memory 32 by means of a base system memory bus 36. The memory controller 28 controls read and write operations to base system memory 32 over the base system memory bus 36, which operations are initiated by either the CPU 24 over the CPU local bus 34, or by an S-bus device 14 over the S-bus 16. Because the memory controller has the capability to manage operations on two buses, operations over the base system memory bus 36 and the CPU local bus 34 may be managed simultaneously. The CPU local bus 34, the base system memory bus 36, and the S-bus are 32-bit buses, each of which buses comprises data, address and control information paths ("D", "A", and "C" in Figure 1) as is typical of such buses.

Base system memory 32 provides system-wide storage capability and may comprise either non-interleaved or interleaved memory cards. The CPU cache 30 permits short term storage of information contained within either base system memory 32 or expansion memory located elsewhere within the system 10. Such expansion memory could, for example, be located on the peripherally attached I/O devices within the system. The CPU cache 30 incorporates random access memory (RAM, not shown) which is used to temporarily stores address locations of the base system memory 32 which are frequently accessed by the CPU 24. The CPU 24 accesses information stored in the CPU cache 30 directly, whereas access to information stored in the base system memory 32 must be handled by the memory controller 28.

All access to base system memory 32 is controlled by the memory controller 28 via base system memory bus 36. The memory controller initiates system memory cycles to the base system memory 32, during which cycles either the CPU 24 or one of the S-bus devices 14 has access to the base system memory via the memory controller 28. During a memory cycle, the memory controller does not pass information onto the S-bus. However, if the memory controller determines that the operation it is managing is an I/O cycle, the memory controller propagates the information onto the S-bus for access thereto by an S-bus device. If the I/O cycle is destined for a S-bus device, the appropriate S-bus device responds with a decode command to the memory controller. If the I/O operation is destined for a primary PCI device

18, the PCI host bridge 20 responds with a decode command to the memory controller and passes the I/O cycle to the appropriate primary PCI device.

A system clock module 38 provides a single clock signal for the S-bus devices 14, and a pair of clock signals for the CPU 24. The CPU 24 requires two clock signals because it operates internally at a higher rate, but communicates over the CPU local bus 34 at a lower rate.

Communications between the processor, cache and memory complex 12 and the S-bus devices are managed by the memory controller 28 over the S-bus 16. Also attached to the S-bus, as shown in the preferred embodiment of Figure 1, are a direct memory access (DMA) controller 40, a system arbitration control point (SACP) 42, an input/output (I/O) controller 44 and a power management controller 48. An optional power management controller 50 may be attached to the power management controller 48 in case more sophisticated power management control is desired. A buffer 52 is provided on the S-bus 16 intermediate the DMA controller 40 and the I/O controller 44. As shown in Figure 1, however, it is contemplated that other S-bus devices 14, beyond those shown, may be attached to the S-bus 16.

Attached to an I/O controller 44 are a time-of-day clock 60 and a RAM module 62. The I/O controller 44 supports a variety of ports, including a mouse port 64, serial ports 66, a parallel port 68, and a keyboard port 70.

In addition to supporting S-bus devices 14 on the S-bus 16, the system 10 also supports a second high speed, high bandwidth bus, which in the preferred embodiment is the primary PCI bus 22. The PCI bus is described in "PCI Local Bus Specification, Revision 2.2, December 18, 1998", published by the PCI Special Interest Group. Primary PCI devices 18 in the system 10 communicate with each other over the primary PCI bus 22. Primary PCI devices communicate with the CPU, cache and memory complex 12 and with other S-bus devices 14 residing on the S-bus 16 by means of the PCI host bridge 20, which is itself an S-bus device residing on the S-bus. The PCI host bridge 20, then, serves as an interface between the S-bus 16 and the primary PCI bus 22 and provides an effective means of communication between these two buses, and any peripheral devices which may be attached to these buses.

The PCI host bridge 20 is a low latency interconnect mechanism through which the CPU 24 or other S-bus device 14 may directly access the primary PCI devices 18 or devices attached thereto. The bridge 20

also provides a high performance path which allows the primary PCI devices or devices attached thereto quick and direct access to base system memory 32. In addition, the host bridge 20 provides all of the hardware required to provide an interface between the S-bus 16 and the primary PCI bus 22 so that data may be transferred between these buses.

The primary PCI bus 22 is capable of supporting a variety of devices which are PCI compatible. As shown in Figure 1, these devices may include a graphics controller 72, a serial SCSI (small computer systems interface) controller 74, a standard I/O bus (for example, ISA bridge 78 (also referred to herein as an expansion bridge)), and a PCI secondary bridge 80. The devices shown in Figure 1 attached to the primary PCI bus, however, are only one example of a system implementing a PCI bus architecture and thus the disclosed exemplary configuration and is not intended to limit the invention in any way.

The graphics controller 72 is typically provided with memory capability in the form of VRAM 82, which enables the graphics controller to buffer video frames therein, and may control any known graphics package which may be supported by PCI bus architecture. The SCSI controller 74 serves as an interface between SCSI devices 84 attached to a SCSI bus 86 and the primary PCI bus 22, and may control any SCSI device which may be supported by PCI bus architecture.

The standard bus bridge 78 serves as an interface between I/O devices 90 attached to a standard (for example, ISA) bus 92 and the primary PCI bus 22. Secondary PCI devices 94 are connected to PCI secondary bridge 80 via secondary PCI bus 96. Any number of unidentified secondary PCI devices 94 may then be connected to the secondary PCI bus 96. The PCI secondary bridge 80 serves as an interface between the secondary PCI devices 94 attached to the secondary PCI bus 96, and the primary PCI bus 22.

Data is communicated between PCI devices which reside on PCI buses in the system over those PCI buses (for example, primary PCI bus 22 and secondary PCI bus 96). Data transactions include read or write operations initiated by a PCI device acting as a master on the bus to or from a PCI device acting as a slave (target) on the bus.

In the manufacture of personal computers, server computers and the like, a customer's order is transmitted to the manufacturing location through a variety of shop floor control systems. The order specifies the type of machine to be built, the operating system to be installed on the machine and any additional hardware items to be added to the personal computer. Once accepted the order is broken down into

software and hardware items. The software items will typically include an operating system, and the hardware will typically be based around a standard personal computer system board with built in adapters, and other additional installable devices, for example, audio, video, network and modem cards among others devices. The customer configuration will be built into a control file that will contain details of all the items for that specific customer. This is used as a master control file for the build of the personal computers and is sent to the manufacturing location. These items effectively specify the bill of material for the personal computer. The various parts are taken from stock and added into the target personal computers. The complete personal computer is then customised.

Figure 2 shows the software portions that are installed onto a prior art personal computer such as that of figure 1 using the installation method of the present invention. First operating system 202 is installed. Typically this may be an operating system such as Windows 95 from Microsoft, the Linux operating system or OS/2 from IBM Corporation (Windows and Microsoft are trademarks of Microsoft Corp. and OS/2 and IBM are trademarks of IBM Corp.). Personalisation files 204 are then installed. These personalisation files include such files as define the language of text to be displayed on the screen to the user. Application files 206 are installed which may include an "office suite" application such as Lotus Smartsuite from Lotus Development Corp. or Microsoft Office (Lotus and Smartsuite are trademarks of Lotus Development Corp.). Also installed are operating system extensions 208 and device drivers 210 for hardware additions to the basic personal computer including such devices as audio devices, video devices, communications devices and mass storage devices.

Whilst in the manufacturing process, various procedures are used on the machine to test and verify the function of the personal computer. During this test and verification process certain information can be obtained pertaining to devices that have been added to the personal computer. In the present invention, system level bus interrogation of the PC system bus will yield a level of data that is specific to each device on the system bus. Since the identification of such devices is controlled by defined standards, which are available in the public domain, every device can be interrogated and their unique configuration identifiers can be read on a real time basis. Using this data as a reference point to identify each device allows the customisation process to act upon this data.

In the PCI bus architecture, each PCI device is provided with configuration space consisting of 256 bytes of register space. This space is divided into a predefined header region and a device

dependent region. The configuration space of each peripheral device contains the data used by the overall configuration software in the system to create an address map. Accordingly, when the CPU 122 initially runs the configuration software, it accesses and reads or writes to the configuration space of each device or bridge to configure the system and create the address map. The PCI Local Bus Specification requires that a device's Configuration Space must be accessible at all times, not just during the system boot. All PCI devices (except host bus bridges) are required to respond to such Configuration Read and Configuration Write commands.

Figure 3 shows a map of the header region of the configuration space provided by configuration registers in each of PCI devices of Figure 2. The predefined header region consists of fields that uniquely define the device. The first 16 bytes of the header region are defined in the same way for all types of devices. Other bytes of the header region and of the Configuration Space are defined according to the type of device.

Vendor ID register 300 contains a 16 bit code which identifies the manufacturer of the device. These codes are allocated by the PCI Special Interest Group (SIG) in order to ensure uniqueness. 0FFFFh is an invalid value for the Vendor ID. Allocated values include 08086h for Intel, 0104Ch for Texas and 01023h for ATI. Device ID register 302 contains a 16 bit code that identifies the particular device. This code is allocated by the vendor. Examples of allocated codes include 07100h, 07110h, 07111h, 07112h and 07113h for Intel Motherboard functions of CPU bridge, PCI bridge, IDE system, USB and PCI bridge as well as 0AC16h for Texas Cardbus resource and 09397h for ATI Video resource. Revision ID Register 308 contains a 16 bit code which identifies the revision or version of the particular device. This code is allocated by the vendor. Header Type register 314 identifies the layout of the second part of the pre-defined header and whether or not the device contains multiple functions. Bit 7 of this register is used to identify a multi-function device. A 16 bit code of 00h corresponds to a PCI device, 01h corresponds to PCI-to-PCI bridges and 02h corresponds to CardBus bridges.

The second part 316 of the pre-defined header for a PCI device includes six 4 byte Base Address Registers located at addresses 010h, 014h, 018h, 01Ch, 010h and 024h.

In order to access one of the Configuration Registers, a "Configuration Read" or a "Configuration Write" Bus Command is sent from the bus master to the device containing the Configuration Register which it is desired to access. A "Configuration Read"

command is indicated by placing 1010 on the Bus Command and Byte Enables (C/BE[3::0]#) pins. A "Configuration Write" command is indicated by placing 1011h on the Bus Command and Byte Enables (C/BE[3::0]#) pins. The IDSEL pin of the agent being accessed is asserted and Address and Data pins 1 and 0 (AD[1::0]) are set to 00h to indicate a Type 0 configuration access to select a device on the bus where the master is located. During the address phase of the Configuration Register access, Address and Data bits 2 to 7 (AD[7::2]) indicate which of the 64 double-word registers in the configuration space of the device are to be accessed. Address and Data bits 8 to 10 (AD[10::8]) indicate which device of a multi-function agent is being accessed. Address and Data bits 11 to 31 (AD[31::11]) are logical don't care states.

In a variation of the access described above, in which it is desired for a configuration request to be passed to another bus segment, Address and Data pins 1 and 0 (AD[1::0]) are set to 01h to indicate a Type 1 configuration. During the address phase of the Configuration Register access, Address and Data bits 11 to 15 (AD[15::11]) indicate which of 32 possible devices on a given bus is being accessed. Address and Data bits 16 to 23 (AD[23::16]) indicate which of 256 possible busses in a system are being accessed.

During the test and verify process the particular hardware specific to the personal computer being customised is identified. The process of identifying the built-in hardware adaptors yields essential data pertaining to the individual devices within the system. This process will now be described, with reference to figure 4. A typical hardware scan will provide information, for each function of each device on each bus, in the following format:

PCI CONFIGURATION REGISTERS FOR BUS 0, DEVICE 0, FUNCTION 0

00	Vendor ID	= 8086
02	Device ID	= 7100
04	Command	= 0006
06	Status	= 2200
08	Revision ID	= 01
09	Programming I/F	= 00
0A	Subclass Code	= 00
0B	Class Code	= 06
0C	Cache Line Size	= 00
0D	Latency Timer	= 20
0E	Header Type	= 00
10	Base Addr Reg 0	= 00000000
14	Base Addr Reg 1	= 00000000

18	Base Addr Reg 2	= 00000000
1C	Base Addr Reg 3	= 00000000
20	Base Addr Reg 4	= 00000000
24	Base Addr Reg 5	= 00000000
28	Cardbus CIS Ptr	= 00000000
2C	Subsystem ID	= 00000000
30	Expansion ROM	= 00000000
34	Reserved	= 00000000
38	Reserved	= 00000000

The data that is read back from the device is of a standard format. The fields of the acquired data that are used in the present invention are as follows:

Subset 00	=	Hardware supplier	(eg Intel)
Subset 02	=	Device type	(eg Intel BX chipset)
Subset 08	=	Revision level	(Revision level of the hardware interrogated)

The process of determining the hardware installed in the personal computer and of installing the relevant software drivers starts at step 402 of figure 4. At step 404, the first bus, first device is interrogated in order to identify the hardware supplier, device type and revision level. At steps 406 and 408, the acquisition process described above is repeated for every possible PCI "slot". The Vendor ID is first read. If the Vendor ID is 0FFFFh, then an invalid vendor ID is indicated and so there is no device installed in that slot. **< How does the software that interrogates the slots know how many slots there may be? - does it interrogate bus 0, slot 0, then slot 1 until it finds an empty slot, then move on to bus 1 etc. until it finds a bus that is not present? For this to work, cards would have to be in the lowest slot available for each bus. >** By multiple interrogations of the hardware for each device and function, a complete copy of the hardware configuration registers is acquired. Once the data has been successfully acquired, at step 410, the data is then split into data relevant to the specific subsytems, for example, motherboard chipset, video chip set and the like as each of these subsytems has specific unique software drivers associated with them. All of these unique hardware associated software sets are already installed on a code server associated with the manufacturing process. **< All buses, all devices - How do you know when complete? >**

The operating system required by the customer is installed on the personal computer using a prebuilt/ready to run image. The operating system is in a format that can be copied directly onto the target personal computer. The operating system is specific to the

type of motherboard installed in the particular personal computer. Without this hardware/association then an inoperative operating system could be copied to the personal computer.

Once the basic operating system has been copied to the target personal computer, the personal computer is prepared for the copying of additional software associated with the hardware specific to that personal computer. The additional software for the additional hardware can only be copied once the operating system directory structure has been established. At this stage a scan is done of the personal computer hardware to determine the exact hardware configuration of the personal computer in question by the use of PCI Configuration Register information. A typical scan of a personal computer would yield the following data, with each device/function requiring an associated set of software.

PCI CONFIGURATION REGISTERS FOR BUS 0, DEVICE 0, FUNCTION 0

```
00  Vendor ID       = 8086    >>> Intel
02  Device ID       = 7100    >>> Motherboard resource (CPU bridge)
08  Revision ID     = 01
```

PCI CONFIGURATION REGISTERS FOR BUS 0, DEVICE 1, FUNCTION 0

```
00  Vendor ID       = 8086    >>> Intel
02  Device ID       = 7110    >>> Motherboard resource (PCI bridge)
08  Revision ID     = 01
```

PCI CONFIGURATION REGISTERS FOR BUS 0, DEVICE 1, FUNCTION 1

```
00  Vendor ID       = 8086    >>> Intel
02  Device ID       = 7111    >>> Motherboard resource (IDE system)
08  Revision ID     = 01
```

PCI CONFIGURATION REGISTERS FOR BUS 0, DEVICE 1, FUNCTION 2

```
00  Vendor ID       = 8086    >>> Intel
02  Device ID       = 7112    >>> Motherboard resource (USB)
08  Revision ID     = 01
```

PCI CONFIGURATION REGISTERS FOR BUS 0, DEVICE 1, FUNCTION 3

```
00  Vendor ID       = 8086    >>> Intel
02  Device ID       = 7113    >>> Motherboard resource (PCI bridge)
08  Revision ID     = 01
```

PCI CONFIGURATION REGISTERS FOR BUS 0, DEVICE 2, FUNCTION 0

```
00  Vendor ID       = 104c    >>> Texas
02  Device ID       = ac16    >>> Cardbus resource
08  Revision ID     = 01
```

PCI CONFIGURATION REGISTERS FOR BUS 0, DEVICE 2, FUNCTION 1

00 Vendor ID = 104c >>> Texas
02 Device ID = ac16 >>> Cardbus resource
08 Revision ID = 01

PCI CONFIGURATION REGISTERS FOR BUS 0, DEVICE 3, FUNCTION 0

00 Vendor ID = 1023 >>> ATI
02 Device ID = 9397 >>> Video Resource
08 Revision ID = f3

The data acquired by the scan is used as pointers to the various software sets that are to be copied into the directory structure of the target personal computer as in the following example:

Subsystem	Supplier	Device type	Revision level	Driver filename
Video	1023	9397	f3	ATIXXX0f3.FIL
Cardbus	104c	ac16	01	TEXXXX001.FIL

At step 412 in figure 4, the required drivers (ATIXXX0f3.FIL and TEXXXX001.FIL in the above example) are identified from the vendor, device and revision information. At step 414, the identified relevant drivers are installed and the process is completed at step 416.

< What does the following sentence from the disclosure mean? >

Expansion of this type of look up table/matrix would allow autoinstall of a variety of hardware subsystems to be built remotely.

Whilst the system above has been described in relation to the installation of operating system software extensions and device drivers for specific pieces of hardware, the present invention could also be applied to the installation of software extensions and device drivers for specific pieces of application software. For example, a word processor could check whether a device for the recognition of voice commands such as the hardware to support a microphone input is installed and if the device is installed, then suitable drivers to allow voice input to the word processor could be automatically installed.

Whilst the invention has been described by way of a preferred embodiment, various modifications and improvements will occur to those person skilled in the art. Therefore it should be understood that the preferred embodiment has been provided as an example and not as a limitation.

CLAIMS

1. A method of installing software associated with hardware installed in a computer, the method comprising the steps of:
providing one or more unique identifiers in the hardware;
interrogating the unique identifiers;
responsive to the interrogation, identifying the software associated with the installed hardware; and
installing the associated software.

2. A method as claimed in claim 1 wherein the unique identifiers are configuration registers.

3. A method as claimed in claim 2 wherein the configuration registers comprise information relating to the vendor, device type and revision level.

4. A method as claimed in claim 3 wherein the associated software is identified by means of vendor, device type and revision level information.

5. A computer software installation system for the installation of software associated with one or more pieces of hardware installed in a computer, Each piece of hardware having a unique identifier, the system comprising:
means for interrogating the unique identifier;
means for identifying which software is associated with the particular piece of hardware; and
means for installing the software corresponding to the particular piece of hardware onto the computer.

6. A system as claimed in claim 5 wherein the unique identifiers are configuration registers.

7. A system as claimed in claim 6 wherein the configuration registers comprise information relating to the vendor, device type and revision level.

8. A system as claimed in claim 7 wherein the associated software is identified by means of vendor, device type and revision level information.

9. A computer program product for use in a data processing system having a non-volatile storage medium, the computer program product comprising:

a computer usable medium having computer readable program code means embodied in said medium for installing software associated with hardware installed in a computer, each piece of hardware having one or more unique identifiers in the hardware, said computer program product having:

computer readable program code means for interrogating the unique identifiers;

computer readable program code means, responsive to the computer readable program code means for interrogation, for identifying the software associated with the installed hardware; and

computer readable program code means for installing the associated software.

10. A system as claimed in claim 9 wherein the unique identifiers are configuration registers.

11. A system as claimed in claim 10 wherein the configuration registers comprise information relating to the vendor, device type and revision level.

12. A system as claimed in claim 11 wherein the associated software is identified by means of vendor, device type and revision level information.

Application No: GB 9930283.8
Claims searched: 1-12

Examiner: Matthew J. Tosh
Date of search: 15 December 2000

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.R): G4A (AFL)

Int Cl (Ed.7): G06F 9/445

Other: ONLINE: EPODOC, WPI, JAPIO, ELSEVIER, INSPEC, TDB

Documents considered to be relevant:

Category	Identity of document and relevant passage	Relevant to claims
A	GB 2329052 A (DELL USA). See whole document.	
A	US 5894571 (DELL USA). See whole document.	

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.